Computers and Mathematics with Applications 63 (2012) 325–336

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A novel parallel hybrid intelligence optimization algorithm for a function approximation problem

Wu Deng [a,b,c,d,e,*], Rong Chen [a], Jian Gao [a], Yingjie Song [a], Junjie Xu [a]

[a] *Informational Science and Technology Institute, Dalian Maritime University, Dalian 116026, China*

[b] *Software Institute, Dalian Jiaotong University, Dalian and 116028, China*

[c] *State Key Laboratory of Power Transmission Equipment & System Security and New Technology, College of Electrical Engineering, Chongqing University, Chongqing and 400044, China*

[d] *Artificial Intelligence Key Laboratory of Sichuan Province (Sichuan University of Science and Engineering), Zigong and 643000, China*

[e] *Key Laboratory of advanced Design and Intelligent Computing (Dalian University), Ministry of Education, Dalian, 116622, China*

ARTICLE INFO

Article history:
Received 30 September 2011
Received in revised form 15 November 2011
Accepted 15 November 2011

Keywords:
Function approximation problem
Genetic algorithms
Particle swarm optimization
Fuzzy neural network
Parallel hybrid intelligence
Optimization algorithm

ABSTRACT

A novel parallel hybrid intelligence optimization algorithm (PHIOA) is proposed based on combining the merits of particle swarm optimization with genetic algorithms. The PHIOA uses the ideas of selection, crossover and mutation from genetic algorithms (GAs) and the update velocity and situation of particle swarm optimization (PSO) under the independence of PSO and GAs. The proposed algorithm divides the individuals into two equation groups according to their fitness values. The subgroup of the top fitness values is evolved by GAs and the other subgroup is evolved by the PSO algorithm. The optimal number is selected as a global optimum at every circulation which shows better results than both PSO and GAs, then improves the overall performance of the algorithm. The PHIOA is used to optimize the structure and parameters of the fuzzy neural network. Finally, the experimental results have demonstrated the superiority of the proposed PHIOA to search the global optimal solution. The PHIOA can improve the error accuracy while speeding up the convergence process, and effectively avoid the premature convergence to compare with the existing methods.

1. Introduction

A Fuzzy Neural Network (FNN) combines the artificial neural network with fuzzy theory in order to make use of their merits and overcome their respective deficiencies, such as stronger self-learning, associative function, less manual intervention and higher precision of the artificial neural network, as well as easier understanding of the reasoning process [1], better use of expert knowledge and lower sample requirements of fuzzy theory. So the FNN can process the abstract information and be provided with the functions of pattern recognition, function approximation, optimization, associative memory and processing fuzzy knowledge. The FNN is widely applied in theory research and application domain. However, for the neural network, if the network structure is too simple, it will cause lower learning ability, and if the network structure is too complex, it will bring lower network induction. Therefore, the structure and parameters of FNN are optimized which are the current major research works.

In recent years, the particle swarm optimization (PSO) algorithm, genetic algorithms (GAs), ant colony optimization (ACO) algorithm, immune algorithm (IA), and so on, which are based on the biological evolution computation method,

* Corresponding author at: Informational Science and Technology Institute, Dalian Maritime University, Dalian 116026, China. Tel.: +86 411 8622 3607; fax: +86 411 8622 3333.
E-mail address: dw7689@gmail.com (W. Deng).

0898-1221/$ – see front matter
doi:10.1016/j.camwa.2011.11.028

have performed an obvious predominance in solving complex nonlinear optimization problems [2], and some research findings have been obtained in the past several decades. Yi presented an improved PSO-based ANN with simulated annealing technique [3]. Chau presented the adoption of a particle swarm optimization model to train perceptrons in predicting the outcome of construction claims in Hong Kong [4]. Tzeng presented an efficient method to design a fuzzy wavelet neural network (FWNN) for function learning and identification by tuning fuzzy membership functions and wavelet neural networks to improve the function approximation accuracy and general capability of the FWNN system [5]. Kou presented a co-evolutionary particle swarm optimization (CPSO) algorithm to solve global nonlinear optimization problems [6]. Majid used a particle swarm algorithm for solving systems of nonlinear equations [7]. Dong presented a rough set and fuzzy wavelet neural network integrated with a least square weighted fusion algorithm for power transformer fault diagnosis [8]. Li presented a fuzzy optimization method based on principal operations and genetic algorithms [9]. Han presented a presented particle swarm optimization algorithm for solving the uncapacitated MLLS problem with assembly structure [10]. In these evolution computation methods, the PSO and GAs are more frequently used to solve complex optimization problems. GAs is one kind of research algorithm which simulates the parallel, randomicity and adaptability of the biological mechanism of natural selection and natural genetics. It can accomplish the global optimization under the complex, multimodal, nonlinear and non-differentiable space and manifest better learning ability and optimization ability. However, due to the complexity of genetic operation, the network training time increases exponentially with the scale and complexity, is lacking in the effective searching mechanism in the local area, and shows the slowness or stagnancy convergence phenomenon near the optimal solution. The PSO is a global optimization evolutionary algorithm based on the iteration method and results from simulating the social behavior of birds within a flock. The best merit of the PSO is the high efficiency for function optimization. Nevertheless, the PSO easily falls into the local extreme point and slows convergence and particle homogenization, and has poor optimization accuracy.

In order to improve the convergence speed and optimization accuracy, according to the above analysis, we propose a novel parallel hybrid intelligence optimization algorithm (PHIOA) based on PSO and GAs by using a parallel evolution idea [3]. The proposed PHIOA is used to optimize the structure and parameters of the FNN. The PHIOA integrates the current global optimization of PSO for guaranteeing the convergence and the global search of genetic algorithms. The PHIOA increases the searching scope and overcomes the disadvantages of PSO and GAs respectively. Finally, the PHIOA improves the convergence speed, the search success probability and the premature convergence, and avoids the local optimum by the experimental verification and comparison.

2. GAs, PSO and FNN

PSO shares many similarities with evolutionary computation techniques such as ACO. In this section, a brief description of PSO and GAs and a detailed description of FNN are given.

2.1. Genetic algorithms (GAs)

GAs are a kind of stochastic global search algorithm which solve complex problems by imitating processes from natural evolution. Based on survival of the fittest and reproduction, GAs exploit continually newer and better solutions without any pre-assumptions, such as continuity and unimodality [11]. GAs have been successfully applied in many complex optimization problems and show their merits over traditional optimization methods, especially the system with multiple local optimum solutions. GAs evolve a population of candidate solutions. Each solution is usually coded as a binary string called a chromosome. The fitness of each chromosome is evaluated by using a performance function after the chromosome has been decoded. Upon completion of the evaluation, a biased roulette wheel is used to select randomly the pairs of better chromosomes to undergo such genetic operations as crossover and mutation that mimic nature. The newly produced chromosomes turn out to be stronger than the weaker ones from the previous generation and should replace these weaker chromosomes. This evolution process continues executing until the stopping criteria are reached [12]. GAs are composed of the encoded mechanism, fitness function, genetic operators, and control parameters, etc. The computation flow is shown in Fig. 1.

2.2. Particle swarm optimization (PSO)

PSO is an evolutionary computation technique that is developed by Kennedy and Eberhart [13]. The PSO concept is based on a metaphor of social interaction such as bird flocking. PSO consists of a number of individuals refining their knowledge of the given search space. The individuals in PSO have a position and a velocity. The PSO algorithm works by attracting the particles to search space positions of high fitness. Each particle has a memory function, and adjusts its trajectory according to two pieces of information, the best position visited, and the global best position attained by the swarm. If the swarm is considered as a society, the first piece of information can be seen as resulting from the particle's memory of its past position, and the second piece of information can be seen as resulting from the collective experience of all members of the society. Like other optimization methods, PSO has a fitness evaluation function that takes each particle's position and assigns a fitness value. The best position visited by the swarm is called the global best. The global position has been personally visited. Each particle is regarded as a point in a N-dimensional space. It tries to modify its position by the concept of velocity. The traits

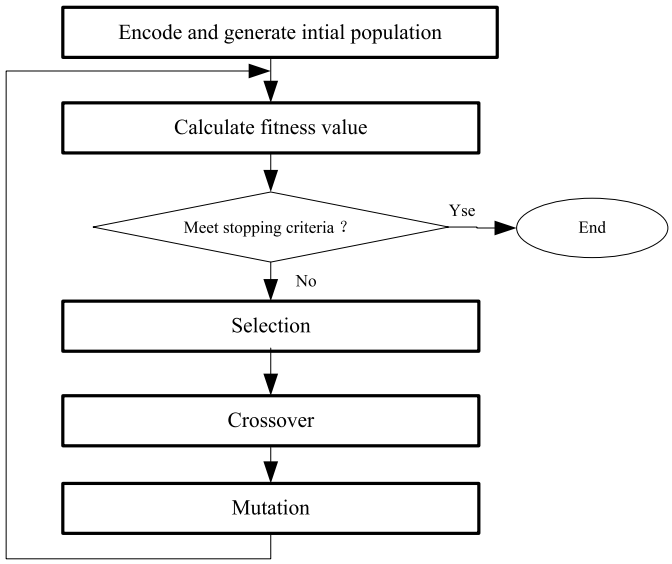

Fig. 1. Calculation flow chart of GAs.

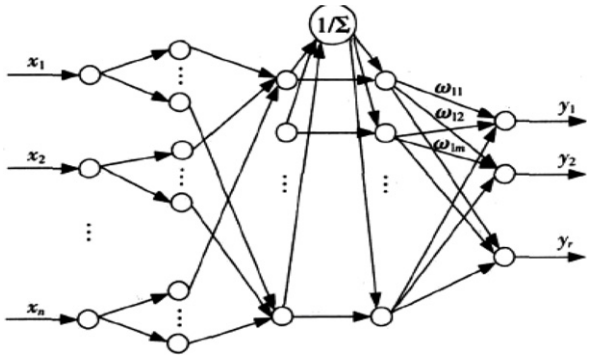

Fig. 2. Fuzzy neural network structure.

of PSO are as follows [14]: (1) PSO is a history based algorithm such that in each step, particles use their own behavior associated with the previous iterations. (2) PSO is easy to implement and, only the parameters to be adjusted. Therefore, the computation takes less time and requires less memory. (3) PSO can solve nonlinear optimization problems with both continuous and discrete variables.

2.3. Fuzzy neural network (FNN)

Fuzzy logic and neural network are based on the mathematical model of dynamic characteristics. Fuzzy logic describes regular expert knowledge, experience or operational data. A Neural network is used to train sample data. FNN is composed of five layers of multi-input and single-output. The general network structure is shown in Fig. 2.

The functions per layer are explained as follows [15]:

First layer (Input Layer). Input vectors in this layer may be either accurate numerical vector or fuzzy value. The '*n*' is the number of nodes which corresponds to the number of input variables, the output of nodes equals the value of the input variable. The formula is shown as:

$$O_i^1 = I_i = x_i, \quad i = 1, 2, 3, \ldots, n. \tag{1}$$

Second layer (Fuzzification Layer). The function is to fuzzify the input signals. In this layer, it can not only implement the membership function of input variables, but also match the control rules of fuzzy control. The number of nodes equals all the possible fuzzy rules through input variables. The output of nodes corresponds to the product of each Gaussian function. The formula is shown as:

$$\mu_{ij} = \exp\left[-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}\right], \quad i = 1, 2, \ldots, r;\ j = 1, 2, \ldots, u \tag{2}$$

where, μ_{ij} is the ith Gaussian function of the jth neuron; c_{ij} is the ith Gaussian function center of the jth neuron; σ_{ij} is the ith Gaussian function standard deviation of the jth neuron; r is the number of input vector (dimension), u is the number of the neuron. The output of the jth neuron is:

$$O_j^2 = \exp\left[\sum_{i=1}^{r} \frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}\right], \quad j = 1, 2, \ldots, u. \tag{3}$$

The third layer (Fuzzy Membership Layer). Each neuron expresses one fuzzy rule in this layer. The input of membership grade expresses the former rule, the reasoning adopts fuzzy contains product reasoning to calculate the incentive intensity of each fuzzy rule, shown as:

$$O_j^3 = w_{a1}O_1^2 \times w_{a2}O_2^2 \times \cdots \times w_{au}O_u^2 = \prod_{j=1}^{u} w_{aj}O_j^2. \tag{4}$$

The fourth layer (Anti-Fuzzification Layer). This layer implements the clear output of the fuzzy neural network. The anti-fuzzification algorithm is provided with the global approximability. This paper adopts the fuzzy method of the gravity solution, shown as:

$$O_j^4 = \frac{O_j^3}{\sum_{j=1}^{u} O_j^3}, \quad j = 1, 2, \ldots, u. \tag{5}$$

The fifth layer (Output Layer). This layer implements the precision calculation, which is

$$y = \sum_{j=1}^{u} w_{bj}O_j^4, \quad j = 1, 2, \ldots, u \tag{6}$$

where, w_{bj} is the connection weight of FNN.

3. Parallel hybrid optimization algorithm (PHIOA) based on PSO and GAs

Without loss of generality, the function optimization problem can be expressed as:

$$\min f(x) = f(x_1, x_2, x_3, \ldots, x_n) \in S,$$

where S denotes n-dimensional search space of $x_i^{\min} \le x_i \le x_i^{\max} (i - 1, 2, 3, \ldots, n)$ where $x_i^{\max}$ and $x_i^{\min}$ denote respectively the upper and lower bound of optimization variable x_i.

PSO is based on social adaptation of knowledge for working, and all individuals are considered to be of the same generation. On the contrary, GAs are based on evolution from generation to generation for working, so in a single generation, the individuals' changes are not considered. PSO and GAs are very similar in their inherent parallel characteristics, whereas experiments show that they have their specific advantages for solving different problems [16]. In order to utilize and integrate their merits and overcome their deficiencies, we propose a parallel hybrid optimization algorithm that combines the evolution ideas of GAs and PSO based on the compensatory property in this paper. The description figure of the proposed algorithm is shown in Fig. 3.

This approach integrates the concept of evolving individuals of GA with the concept of self improvement of PSO, where individuals are enhanced by social interactions and their private cognition. The algorithm initializes with a population of random solutions and searches optimized solving.

The steps of the PHIOA are described as follows [17–19]:

Step 1. (Initialization)

In allusion to the N-dimensional solving problem, we need determine the size of the population. In this paper, $4N$ individuals are generated randomly. These individuals may be regarded as particles in the case of PSO and chromosomes in the case of GAs.

Step 2. (Computation and update)

The fitness values of all individuals in the population are computed. Then the population is divided into two subgroups with equal individuals according to the computed fitness value.

Step 3. Application GAs

The GAs are a numerical optimization algorithm by simulating the biological nature selection and genetic mechanism of the colony. It gradually approaches the global optimal solution through the repeated evolution and iteration, based on fitness function for selection, crossover, and mutation. A real coding method is used in this paper, the crossover probability P_c and mutation probability P_m.

Step 3.1 Selection

Selection is an operator to select two parent strings for generating new strings. In the selection, we select the top $2N$ individuals from $4N$ individuals for constructing one subgroup. In GAs, parent strings are selected by random choice.

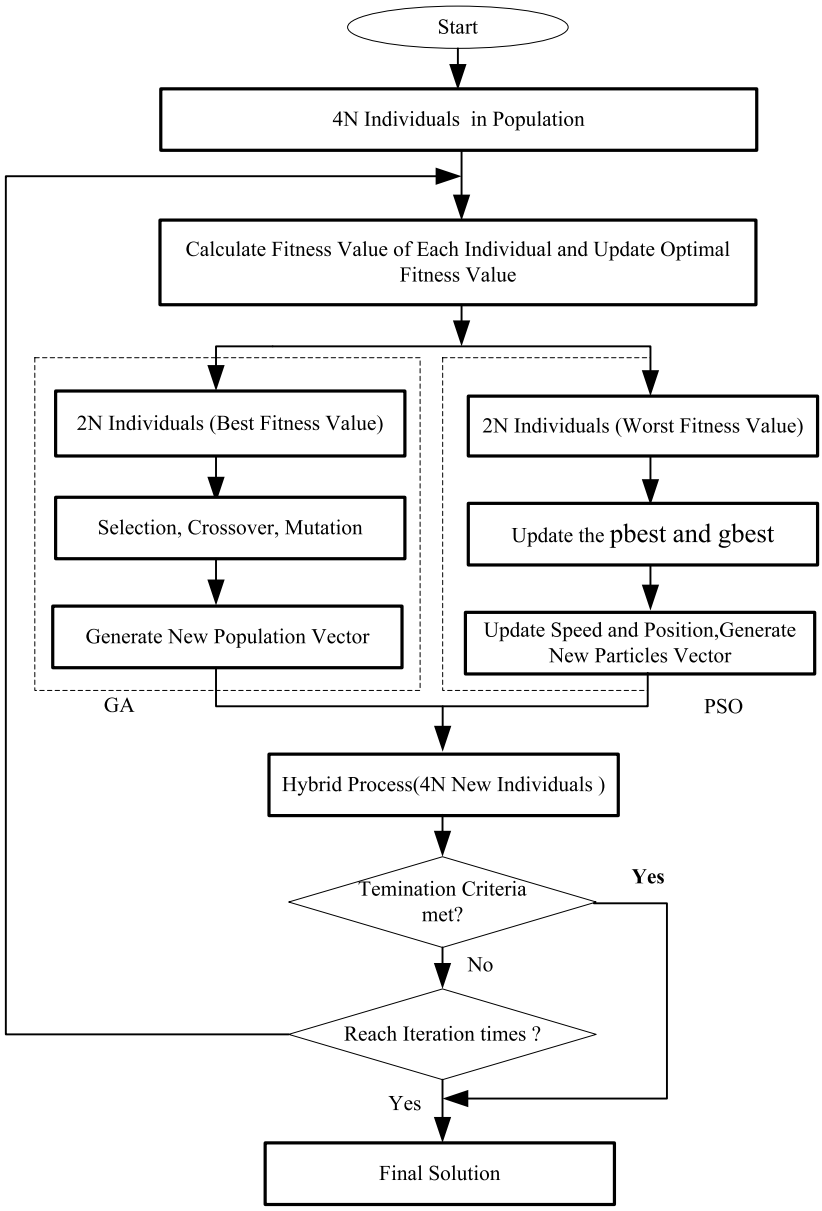

Fig. 3. Parallel hybrid optimization algorithm PHIOA.

The parent strings, however, are not selected by a sheer random choice. The fitness value of each string is utilized for selecting parent strings.

Step 3.2 Crossover

Crossover is an operator to generate new strings from parent strings. The best $2N$ individuals are matched respectively by pairs and crossed with the random according to the crossover probability (P_c). Crossover operation exchanges the position of two encoded strings in order to obtain next-generation encoded strings. There are many crossover operation methods. In this paper, the parent crossover method is selected, the formula is shown as:

$$x_i' = p_i x_i + (1 - p_i)x_{i+1} \quad i = 1, 2, 3, \ldots, 2N - 1 \tag{7}$$

$$x_i' = p_i x_i + (1 - p_i)x_1 \quad i = 2N. \tag{8}$$

In formula (7) and (8), p_i denotes the proportional distributed random number between 0 and 1. x_i denotes the position of the ith particle.

Step 3.3 Mutation

Mutation is an operator to change elements in a string which is generated by a crossover operator. The best 2*N* individuals implement the mutation operation with one mutation probability. The mutation formula is shown as:

$$x'_k = x_k + rand \times N(0, 1). \tag{9}$$

Step 4. Application PSO [20]

The bottom 2*N* individuals are selected from 4*N* individuals for constructing one subgroup to feed into PSO.

Step 4.1 Update speed

The particle's velocity is updated according to the equation below.

$$v_{id}^{new} = w \times v_{id}^{old} + c_1 \times rand_1 \times \left(pbest_{id} - x_{id}^{old}\right) + c_2 \times rand_2 \times \left(gbest_{gd} - x_{id}^{old}\right) \tag{10}$$

where $i = 1, 2, 3, \ldots, N$, and N is the size of the population; c_1 and c_2 are two positive constants, called the cognitive and social parameter respectively (acceleration coefficients) which are used to adjust the own experience and social experience of particles in the motion in order to get the individual optima or the global optima. If $c_1 = 0$, the particle only has 'self-experience', its convergence rate may be fast, and it is easy to fall into the local optimum. If $c_2 = 0$, the particle only has 'social experience', all particles in a swarm become moving by themselves without interaction, and the probability of finding a solution is very little. If $c_1 = c_2 = 0$, then the particle has no 'experience', therefore all particles in a swarm become disorderly and unsystematic. In general, the value of c_1 and c_2 is [0, 4], we select $c_1 = c_2 = 2$ in this paper. $rand_1$ and $rand_2$ are the random numbers evenly distributed within the range [0, 1].

w is an inertial weight which denotes the influence of the previous velocity of a particle upon its current velocity. If $w = 0$, PSO will shrink the current global best position, like a local algorithm. If $w \neq 0$, the particles have extended the trend of the search space, that is global search ability. Therefore, the bigger the w, the bigger the velocity v_{id}, and the bigger the search space for the particles, which helps find new solution spaces. The smaller the w, the smaller the velocity v_{id}, which helps find a better solution in the current solution space, the computation formula is shown as:

$$w = w_{\max} - (w_{\max} - w_{\min}) \times I / I_{\max} \tag{11}$$

where $w_{\max}$ is the initial inertia weight of the velocity, $w_{\min}$ is the final inertia weight of the velocity. I is the current iteration time, $I_{\max}$ is the total iteration time.

Step 4.2 Update position

The particle's position is updated according to the equation below.

$$x_{id}^{new} = \begin{cases} x_{id}^{old} + v_{id}^{new} & x_{\min i} \leq x_{id}^{new} \leq x_{\max_i} \\ x_{\min i} & x_{id}^{new} < x_{\min_i} \\ x_{\max i} & x_{id}^{new} > x_{\max_i}. \end{cases} \tag{12}$$

This equation provides the new position of the particle x_{id}^{new}, adding its new velocity v_{id}^{new} to its current position x_{id}^{old}.

Step 4.3 Update the part best value (pbest)

For each particle, its fitness value is compared with the fitness value of best position particle. If the fitness value of the best position particle is good, the fitness value of the best position is regarded as the current best position and pbest value is updated.

Step 4.4 Update the global best particle (gbest)

For each particle, its fitness value is compared with the fitness value of best global position particle. If the fitness value of the best global position particle is good, the fitness value of best global position is regarded as the current best global position and gbest value is updated.

Step 5. Termination criteria

The obtained fitness values are compared in order to obtain the global optimal value gbest. If gbest meets the termination criteria, the hybrid algorithm will accomplish the task. If gbest does not meet the termination criteria, the *Steps* 2–4 will be repeated until the current iteration reaches the predetermined maximum iteration.

4. Optimize fuzzy RBF neural network (FRBFNN) using PHIOA

4.1. Optimize FRBFNN based on PHIOA

The structure of fuzzy radial basis function neural networks (FRBFNN) includes typically the number of neurons per layer and the connection number between neurons of two adjacent layers. For FRBFNN, the number of neurons is to change only in the third layer, and these neurons play an important role. It is expressed by using the connection number of neurons. So the structure optimization of FRBFNN is to determine the number of nodes in the third layer, the connection number and connection weights among the second layer, the third layer and the fourth layer.

4.1.1. Select initial population

A chromosome is a fuzzy neural network structure and its parameter. The initial population contains the corresponding network structure, the individuals of the input variables and output variables, and the remaining individuals are randomly generated [21]. The $4N$ network structures are randomly generated. Each network structure adopts real-valued encoding. Each encoding individual (chromosome) corresponds to a network structure, so the FRBFNN needs to optimize $4N$ network structures. In this paper, the network structure and its corresponding encoding construct an individual in the population to implement the operation of the parallel hybrid algorithm.

4.1.2. Fitness function

The network error function uses mean square error (MSE) and the network complexity uses the number of nodes in the second layer, the third layer and the sum of connections among the second layer, the third layer and the fourth layer to evaluate. The network error optimization and the control network complexity are synchronously taken into account in order to obtain the optimal fuzzy rules. The fitness function of the network is shown as:

$$F(i, t) = \frac{\alpha}{E(i, t)} + \frac{\beta}{H(i, t)} \tag{13}$$

where, $F(i, t)$ is the ith network individual fitness of the tth generation;

$E(i, t)$ is the ith network individual error of the tth generation;

$H(i, t)$ is the ith network individual complexity of the tth generation;

α and β are constant (> 0), and $\alpha + \beta = 1$.

The above function is used as a fitness function in order to guarantee getting the appropriate network structure and optimizing the network weights. At the same time, the values of α and β are determined according to the actual problem, or adjusted adaptively in course of network training.

4.1.3. Genetic operation

Genetic operations include the reproduction, crossover and mutation. The reproduction adopts a roulette wheel scheme to select the individual according to the adaptive value in the previous generation population. The elite-selection heuristic algorithm is used in this paper. The optimal individual is preserved unconditionally to the next generation in each generation population. It guarantees the asymptotic convergence of the algorithm.

In the genetic algorithms, crossover and mutation are the most important genetic operators which can transfer the prominent information to the next generation. The values of p_c (crossover probability) and p_m (mutation probability) depend on the fitness value in algorithm running. The better fitness value and low p_c and p_m can increase the opportunity of the individual to enter the next generation. For the solution of the lower fitness value, the biggest p_c and p_m can speed up elimination. When the mature convergence occurs, we should increase the values of p_c and p_m in order to accelerate the new individual production. In this paper, we use an adaptive method and the individual fitness function for measuring the convergence of this algorithm. The adaptive genetic algorithms are used to generate the next generation subgroup for the individual. The crossover probability and mutation probability are calculated according to the following equation:

$$p_c = \begin{cases} k_1(f_{\max} - f')(f_{\max} - \bar{f}), f' \geq \bar{f} \\ k_3, \quad f' < \bar{f} \end{cases} \tag{14}$$

$$p_m = \begin{cases} k_2(f_{\max} - f)(f_{\max} - \bar{f}), f \geq \bar{f} \\ k_4, \quad f < \bar{f} \end{cases} \tag{15}$$

where $f_{\max}$ is the best fitness value in the population; $\bar{f}$ is the average fitness degree in the population; f' is the bigger fitness value for crossover by using two strings, f is the fitness value of the mutation string; k_1, k_2, k_3, k_4 are constants with the value [0, 1].

Crossover operation only operates genes in the hidden layer. When the control gene implements the crossover operation in the hidden layer, it will take the network weights of the corresponding genes. The crossover operation of the hybrid encoding of the hierarchical structure resets not only controls genes, but also network weights.

Mutation operation includes the mutation of control gene, network weights of control gene, and the encoding condition parameters (m_{ij} and δ_{ij}) with control gene. The mutation of control gene and condition weights will mutate the value of control gene from 0 to 1, or from 1 to 0, conditional membership parameters (m_{ij} and δ_{ij}), and the connection weights (ω_j) between the third layer and forth layer according to the certain probability. In this paper it is solved by evolutionary programming. Their mutation is expressed as:

$$m'_{ij} = m_{ij} + \alpha \frac{1}{f} N(0, 1) \tag{16}$$

$$\delta'_{ij} = \delta_{ij} + \alpha \frac{1}{f} N(0, 1) \tag{17}$$

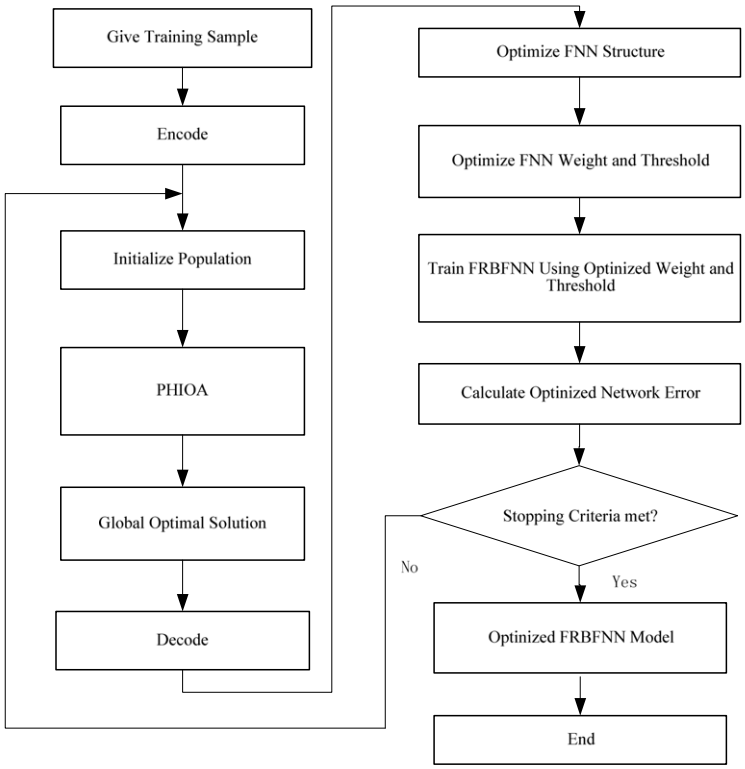

Fig. 4. Training FRBFNN.

$$\omega'_{ij} = \omega_{ij} + \alpha \frac{1}{f} N(0, 1) \tag{18}$$

where α is the evolutionary rate, f is the fitness degree of each individual, $N(0, 1)$ is the expected value (0) and standard deviation of normal distribution random quantity (1).

4.2. Train FRBFNN

The optimal weight threshold imports into the fuzzy neural network to train the network, calculate the expected output and actual network output error. If the error meets the requirements, the network training will end. Otherwise, the weights will continue modifying until the result meets the expectation goal. The training flow chart is shown in Fig. 4.

5. Simulation experiment and result analysis

5.1. The experiment for training FRBFNN

For the training of FRBFNN, these samples were divided into two sets: training samples and testing samples, 300 samples were used for training. GAs, PSO, and the proposed PHIOA were tested and compared in terms of their training performance. The proposed algorithm was coded in MATLAB 2009a. The experiments were conducted on a computer with 2.5 G Pentium (R) Dual-Core CPU E5200, 2.0 G of RAM. Each algorithm was tested 20 times for training FRBFNN, in which 400 generations of training were performed in each trial. The optimum parameters for GAs, PSO and the proposed PHIOA are shown in Table 1.

In order to verify the performance of the proposed PHIOA, FRBFNN was respectively trained with GAs, PSO and PHIOA. The trained results were compared in this paper. The trained time and mean square error (MSE) were summarized in Table 2.

From Fig. 5 and Table 2, three prominent findings can be observed. First, the PHIOA outperformed the other two approaches, especially the PHIOA, which could be reflected by the lowest MSE value. GAs did not perform very satisfactorily when compared with the other two learning algorithms. Secondly, the lowest MSE value, i.e. 0.0973, was trained by the proposed PHIOA due to the combined contribution of the evolutionary mechanism by GAs at every M generations and learning convergence of PSO. This result demonstrated the effectiveness of the proposed PHIOA in training the FRBFNN. Thirdly, for the evolution of 400 generations under each algorithm, the time spent on training the PHIOA was far less than the total time spent on training the GAs and PSO.

Table 1
Parameters for GAs, PSO and PHIOA.

Parameter	GAs	PSO	PHOA
Population size	100	100	100
Code type	Real number	Real number	Real number
Inertia value	N/A	N/A	0.80
Maximum velocity	N/A	0.50	0.50
Learn factor	N/A	$c_1 = c_2 = 2$	$c_1 = c_2 = 2$
Selection operator	Stochastic universal sampling	N/A	Stochastic universal sampling
Crossover operator	Single point	N/A	Single point
Crossover rate	0.90	N/A	0.90
Mutation method	Gaussian	N/A	Gaussian
Mutation operator	Real-value	N/A	Real-value
Mutation rate	0.01	N/A	0.01
Proportion of PHOA	N/A	N/A	0.50

Table 2
The trained time and MSE values of each algorithm under 400 generations.

Algorithm	Training time	MSE
GAs	3 min 46 s	0.5128
PSO	2 min 29 s	0.1067
PHIOA	1 min 17 s	0.0973

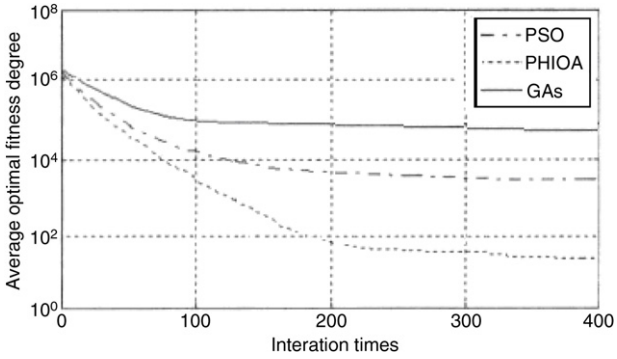

Fig. 5. Training flow chart of FRBFNN.

All these findings indicated that the proposed hybrid algorithm offered exciting advantages over the conventional methods during the learning process of the fuzzy neural network. All the best adjustable parameters of the fuzzy neural network can be obtained by repeated training PHIOA. Consequently, an optimal fuzzy neural network model can be obtained evolutionarily. With the use of the PHIOA, faster convergence of the evolution process to search for an optimal fuzzy neural network can be achieved.

5.2. Simulation experiment

In this section, the performance of the proposed FRBFNN with PHIOA is examined by optimizing approximate nonlinear functions. The given function approximation problem with two inputs and one output is shown as:

$$y = f(x_1, x_2) = \sin(\pi x_1^2) \times \cos(\pi x_2^2) \quad x_1, x_2 \in [-1, 1]. \tag{19}$$

The parameter settings for the proposed PHIOA are shown in Table 1. Let $\sigma_1 = \sigma_2 = 0.4$. According to the configuring method of the FRBFNN center, the number of the centers is obtained, $m_1 = m_2 = 8$, $m = 8 \times 8 = 64$. So we obtain the network center value c_{x_1}, c_{x_2} and the number of hidden nodes of the link layer $p = 4$. In the FRBFNN, the number of weights $w = [w_1, w_2, w_3, w_4]$. That is 4 parameters for each chromosome (or particle) are learned in the FRBFNN.

In order to test the learning validity and verify the generalized ability of the PHIOA, a set of new data different from the trained pattern data and with the equal interval distance 2/19, $\beta = 0.2$ or 0.8, $\gamma = 0.8$ are used as the input to the network which was trained. The simulation result of the generalization ability of the proposed network is shown in Figs. 6–9. Figs. 6 and 7 provide an analysis of learning convergence in terms of generated error in each generation. In two figures, it is noticeable that generation error is declining when generation times are increasing. And the rate of convergence

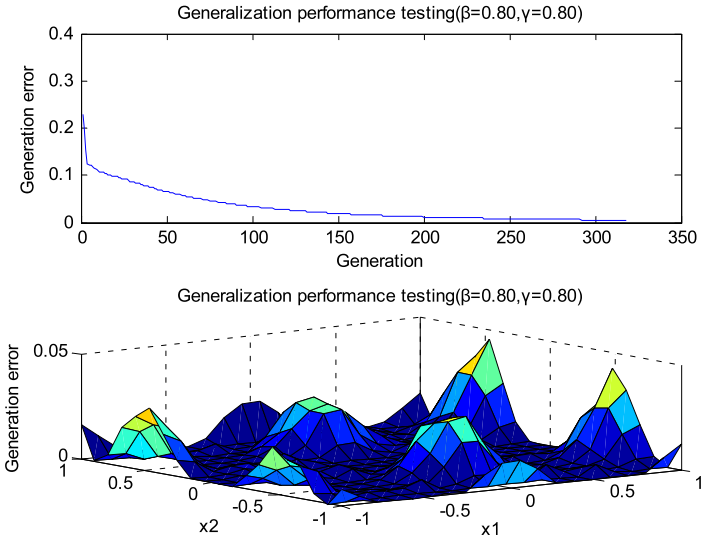

Fig. 6. Generation step and generation error ($\beta = 0.8$, $\gamma = 0.8$).

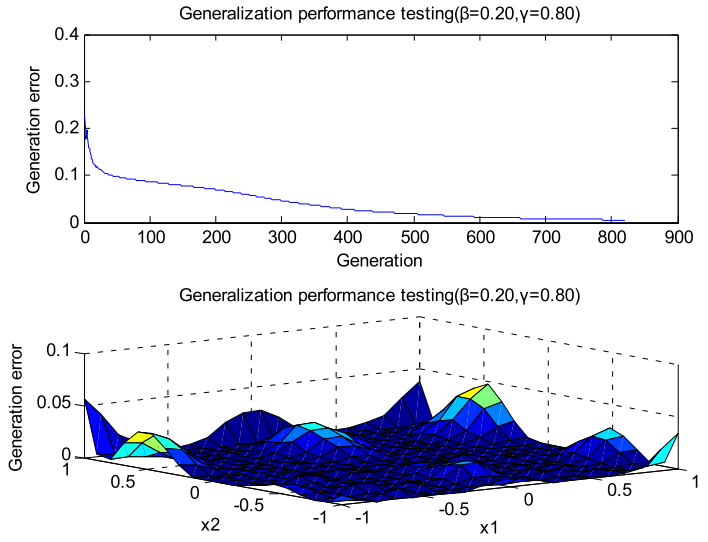

Fig. 7. Generation step and generation error ($\beta = 0.2$, $\gamma = 0.8$).

in the Fig. 7 ($\beta = 0.8$, $\gamma = 0.8$) has a significant effect on the error generated in the Fig. 8 ($\beta = 0.2$, $\gamma = 0.8$). In addition, Figs. 8 and 9 show the dependency of learning convergence on the β value. The expected output and the actual output are compared in Fig. 8 ($\beta = 0.8$, $\gamma = 0.8$) and Fig. 9 ($\beta = 0.2$, $\gamma = 0.8$). From the simulation results of the example, the PHIOA proves the effectiveness in adjusting the weight of the FRBFNN.

6. Conclusion

In this study, a novel parallel hybrid intelligence optimization algorithm named PHIOA is proposed which combines the GAs with PSO for optimizing a fuzzy neural network and solving the function approximation problem. PSO is based on social adaptation of knowledge for working, and all individuals are considered to be of the same generation. The particles with higher degree of constraint violation fly by the search space according to the information exchanged by their pbest and gbest to search the better positions. On the contrary, GAs are based on evolution from generation to generation for working, so in a single generation, the individuals' changes are not considered. PSO and GAs are very similar in their inherent parallel characteristics, whereas experiments show that they have their specific advantages when solving different problems. In the proposed PHIOA algorithm, the PHIOA selects the optimal solution of PSO and GAs in order to improve the performance of PSO and GAs per step for improving the performance of PHIOA. Because the PHIOA transfers the optimal value in the iterative process, it keeps the respective independence and reduces the complexity of PSO and GAs. The main advantages

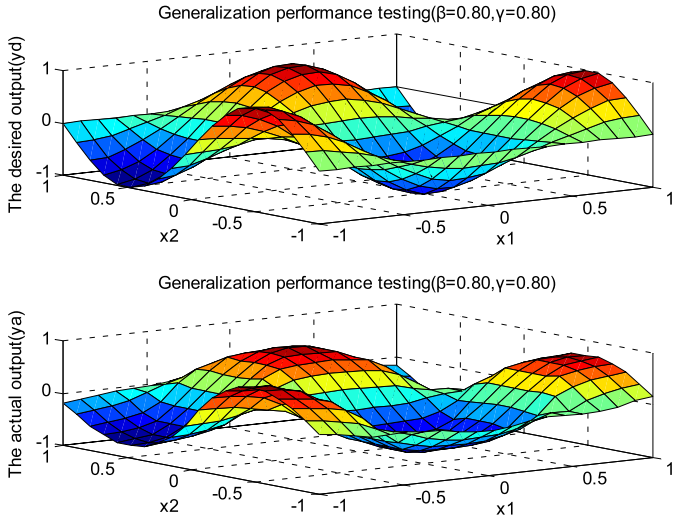

Fig. 8. Comparing of the expected output and the actual output ($\beta = 0.8$, $\gamma = 0.8$).

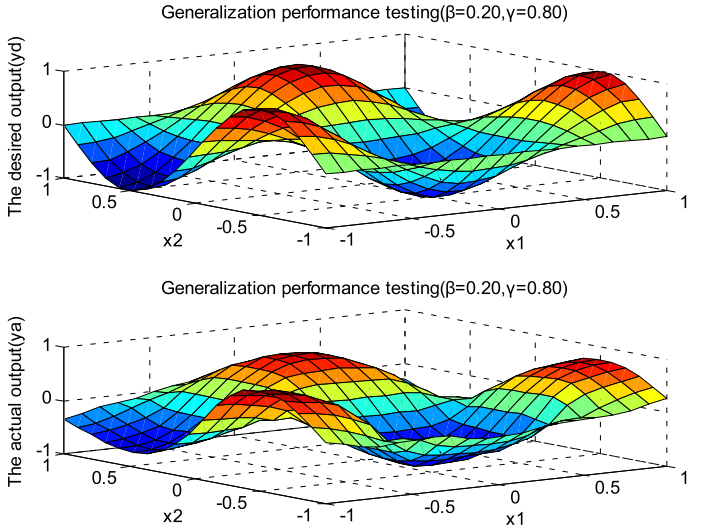

Fig. 9. Comparing of the expected output and the actual output ($\beta = 0.2$, $\gamma = 0.8$).

of the PHIOA over the standard PSO algorithm and GAs is demonstrated within a comparative study and then the function approximation problem has been solved by using the PHIOA algorithm. The results show that the PHIOA is provided with the rapidness convergence, global search capability and good convergence. Future work will be focused in two directions: the application of PHIOA to real industry domain application and the extension of the method to solve the other optimization problems.

Acknowledgments

The authors would like to thank all the reviewers for their constructive comments. This research was supported by the National Natural Science Foundation of China (61175056, 51175054, 60775028, 60870009), the Visiting Scholarship of State Key Laboratory of Power Transmission Equipment & System Security and New Technology (Chongqing University) (2007DA10512711406), the Open Project Program of Artificial Intelligence Key Laboratory of Sichuan Province, (Sichuan University of Science and Engineering), China (2010RZ004), the Open Project Program of Key Laboratory of advanced Design and Intelligent Computing (Dalian University), Ministry of Education, China (ADIC2010008), the Open Project Program of Key Laboratory of Intelligent Manufacture of Hunan Province (Xiangtan University), China (No. 2009IM03), the Open Project Program of Key Laboratory of Numerical Simulation in the Sichuan Provincial College (Neijiang Normal University) China (2011SZFZ001), Natural Science Foundation of Xiangtan University (10XZX16, 11QDZ41), Scientific Research Fund of Hunan

Provincial Education Department (11C1215) and Scientific Research Fund of Hunan Science and Technology Department (2011GK3205). The programme for the initialization, study, training, and simulation of the proposed algorithm in this article was written with the tool-box of MATLAB 2009a produced by Math-Works, Inc.

References

[1] H.Q. Zhao, J.S. Zhang, Functional link neural network cascaded with Chebyshev orthogonal polynomial for nonlinear channel equalization, Signal Processing 88 (8) (2008) 1946–1957.
[2] C.H. Cheng, T.L. Chen, L.Y. Wei, A hybrid model based on rough sets theory and genetic algorithms for stock price forecasting, Information Sciences 180 (9) (2010) 1610–1629.
[3] D. Yi, X.R. Ge, An improved PSO-based ANN with simulated annealing technique, Neurocomputing 63 (2005) 527–533.
[4] K.W. Chau, Application of a PSO-based neural network in analysis of outcomes of construction claims, Automation in Construction 16 (5) (2007) 642–646.
[5] S.T. Tzeng, Design of fuzzy wavelet neural networks using the GA approach for function approximation and system identification, Fuzzy Sets and Systems 161 (1) (2010) 2585–2596.
[6] X.L. Kou, S.Y. Liu, J.K. Zhang, W. Zheng, Co-evolutionary particle swarm optimization to solve constrained optimization problems, Computers & Mathematics with Applications 57 (11–12) (2009) 1776–1784.
[7] J. Majid, K. Esmaile, K. Behrooz, Particle swarm algorithm for solving systems of nonlinear equations, Computers & Mathematics with Applications 62 (2) (2011) 566–576.
[8] L.X. Dong, D.N. Xiao, Y.S. Liang, Y.L. Liu, Rough set and fuzzy wavelet neural network integrated with least square weighted fusion algorithm based fault diagnosis research for power transformers, Electric Power Systems Research 78 (1) (2008) 129–136.
[9] F.C. Li, L.M. Liu, C.X. Jin, Study on fuzzy optimization methods based on quasi-linear fuzzy number and genetic algorithm, Computers & Mathematics with Applications 57 (1) (2009) 67–78.
[10] Y. Han, J.F. Tang, I. Kaku, L.F. Mu, Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight, Computers & Mathematics with Applications 57 (11–12) (2009) 1748–1755.
[11] M.L.D. Wong, A.K. Nandi, Automatic digital modulation recognition using artificial neural network and genetic algorithm, Signal Processing 84 (2) (2004) 351–365.
[12] G.C. Liao, T.P. Tsao, Application of fuzzy neural networks and artificial intelligence for load forecasting, Electric Power Systems Research 70 (3) (2004) 237–244.
[13] J. Kennedy, R.C. Eberhart, Particle swarm optimization [*C*], IEEE International Conference on Neural Networks (1995) 1942–1948. Piscataway: IEEE.
[14] B. Liu, L. Wang, Y.H. Jin, An effective PSO-based memetic algorithm for flow shop scheduling, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 37 (1) (2007) 18–27.
[15] Y. Volkan Pehlivanoglu, Oktay Baysal, Vibrational genetic algorithm enhanced with fuzzy logic and neural networks, Aerospace Science and Technology 14 (2010) 56–64.
[16] K.M. Saridakis, A.J. Dentsoras, Integration of fuzzy logic, genetic algorithms and neural networks in collaborative parametric design, Advanced Engineering Informatics 20 (2006) 379–399.
[17] H.I. Shaheen, G.I. Rashed, S.J. Cheng, Application and comparison of computational intelligence techniques for optimal location and parameter setting of UPFC, Engineering Applications of Artificial Intelligence 23 (2010) 203–221.
[18] N. Taher, A. Babak, An efficient hybrid approach based on PSO, ACO and *k*-means for cluster analysis, Applied Soft Computing 10 (1) (2010) 83–197.
[19] F. Van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation in Evolutionary Computation 8 (3) (2004) 225–239.
[20] W.C. Yeh, A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems, Expert Systems with Applications 26 (2009) 9192–9920.
[21] M.A. Majig, A.R. Hedar, M. Fukushima, Hybrid evolutionary algorithm for solving general variational inequality problems, Journal of Global Optimization 38 (4) (2007) 637–651.